Json

- **J**ava**S**cript **O**bject **N**otion which is commonly known as **JSON** is one of the most popular data transition formats.
- The JavaScript Object Notation (JSON) data format enables applications to communicate over a network, usually through RESTful APIs.
- All modern languages (e.g., Java, JavaScript, Ruby, C#, PHP, Python, and Groovy) and platforms provide excellent support for producing (serializing) and consuming (deserializing) JSON data.
- JSON is very simple as it consists of developer-friendly constructs such as Objects, Arrays, and name/value pairs.
- JSON is not limited to Representational State Transfer (REST); it also works with the following:
    - Node.js (which stores project metadata in package.json)
    - NoSQL databases such as MongoDB
    - Messaging platforms such as Kafka
- It is a text-based and lightweight format for data transactions. JSON format was first computed by Douglas Crockford.
- This being a text-based format is easier to read or write by the user and at the same time, its lightweight property makes it a stress-free alternative for machines to deconstruct or generate.
- It is basically a subset of the JavaScript but JSON, as a text format is totally independent of any of the programming languages used as almost all the languages, can easily analyze the text.
- Its unique properties like text-based, lightweight, language independence etc. make it an ideal candidate for the data-interchange operations.

Features of JSON

- Easy to use – JSON API offers high-level facade, which helps you to simplify commonly used use-cases.
- Performance – JSON is quite fast as it consumes very less memory space, which is especially suitable for large object graphs or systems.
- Free tool – JSON library is open source and free to use.
- Doesn't require to create mapping – Jackson API provides default mapping for many objects to be serialized.
- Clean JSON – Creates clean, and compatible JSON result that is easy to read.
- Dependency – JSON library does not require any other library for processing.

Usage of JSON

JSON is mostly used to transfer the data from one system to another. It can transfer data between two computers, database, programs etc.

- It is mainly used for transmitting serialized data over the network connection.
- It can be used with all the major programming languages.
- It is useful in data transition from the web application to the server.
- Most of the web services use JSON based format for data transfer.

Here are the important benefits/ pros of using JSON:

- Provide support for all browsers
- Easy to read and write
- Straightforward syntax
- You can natively parse in JavaScript using eval() function
- Easy to create and manipulate
- Supported by all major JavaScript frameworks
- Supported by most backend technologies
- JSON is recognized natively by JavaScript
- It allows you to transmit and serialize structured data using a network connection.
- You can use it with modern programming languages.
- JSON is text which can be converted to any object of JavaScript and send to the server.

JSON Object

- JSON object is a set of Keys along with its values without any specific order.
- The key and their values are grouped using curly braces, both opening and closing "{ }".
- So, in the previous *Example* when we were creating a JSON with a car attribute, we were actually creating a JSON car Object. There are certain rules that need to be followed while creating a JSON structure, we will learn about those rules while discussing the Key value pairs.

```
Var chaitanya={
"firstName":"Chaitanya",
"lastName":"Singh",
"age":  "28"};
```

- So, in order to create a JSON, the first thing we will need is an attribute. Here, we are creating an “Employee” JSON object. Next thing we need is to specify the properties of the object, let’s assume our employee have a “First Name”, “Last Name”, “employee ID” and “designation”. These properties of the employee are represented as “Keys” in the JSON structure.

- **Let’s create a JSON object:**

```
{
 "FirstName"            : "Sam",
 "LastName"             : “Jackson”,
 "employeeID"           : 5698523,
 "Designation"          : "Manager",
}
```

- Everything within the curly braces is known as **JSON Employee Object**.
- A basic JSON object is represented by Key-Value pair. In the previous Example, we used a JSON to represent an employee data.
- And we have represented different properties for the employee; “First Name”, “Last Name”, “employee ID” and “designation”. Each of these “keys” has a value in the JSON. For Example, “First Name” has been represented by a value “Sam”. Similarly, we also have represented other keys by using different values.

Generic Rules to be followed while creating a JSON:

- JSON Objects should start and end with braces “{ }”.
- Key fields are included in the double quotes.
- Values are represented by putting “:” colon between them and the keys.
- JSON key-value pairs are separated by a comma “,”.
- Values can be of any data type like String, Integer, Boolean etc.

JSON Arrays

- Arrays in JSON are similar to the ones that are present in any programming language, the array in JSON is also an ordered collection of data.
- The array starts with a left square bracket “[“and ends with right square bracket “]”. The values inside the array are separated by a comma.
- There are some basic rules that need to be followed if you are going to use an array in a JSON.
- Let’s have a look at a sample JSON with an Array.
- We will use the same Employee object that we used earlier.
- We will add another property like “Language expertise”. An employee can have expertise in multiple programming languages. So, in this case, we can

use an array to offer a better way to record multiple language expertisevalues.

```
{
 "FirstName"             : "Sam",
 "LastName"              : "Jackson",
 "employeeID"            : 5698523,
 "Designation"           : "Manager",
 "LanguageExpertise"     : ["Java", "C#", "Python"]
}
```

They are:

- An array in JSON will start with a left square bracket and will end with a right square bracket.
- Values inside the array will be separated by a comma.
- Objects, Key-value pair, and Arrays make different components of the JSON. These can be used together to record any data in a JSON.
- Now, as we have already discussed the basic structure of JSON lets start working on a more complex JSON structure.

Employee JSON

```
{
 "FirstName"             : "Sam",
 "LastName"              : "Jackson",
 "employeeID"            : 5698523,
 "Designation"           : "Manager",
 "LanguageExpertise"     : ["Java", "C#", "Python"]
}
```

JSON (JavaScript Object Notation)	XML
JSON is simple and easier to read and write	XML is verbose and less readable
JSON doesn't use end tag	In XML, the end tag is mandatory
JSON supports array thus it is easy to transfer a big chunk of homogeneous data items using JSON	XML doesn't support array
JSON is easier to parse and can be parsed to ready-to-use JavaScript object	XML is difficult to parse than JSON
JSON is short	XML document is lengthy, verbose and redundant
JSON is less secure than XML	XML is more secured than JSON
JSON file is more readable than XML because it is short and to the point.	XML file is big and filled with user-defined tags, thus less-readable
JSON is data-oriented	XML is document-oriented

JSON object has a type	XML data is typeless
Retrieving value is easy	Retrieving value is difficult
Supported by many Ajax toolkit	Not fully supported by Ajax toolkit
It supports only UTF-8 encoding.	It supports various encoding.

JSON.parse()

- JSON parsing is the process of converting a JSON object in text format to a Javascript object that can be used inside a program.
- In Javascript, the standard way to do this is by using the method JSON.parse(), as the Javascript standard specifies.
- The JSON.parse() method parses a JSON string, constructing the JavaScript value or object described by the string.

JSON.stringify()

- The **JSON.stringify()** method in Javascript is used to create a JSON string out of it.
- While developing an application using JavaScript, many times it is needed to serialize the data to strings for storing the data into a database or for sending the data to an API or web server.
- The data has to be in the form of strings. This conversion of an object to a string can be easily done with the help of the JSON.stringify() method.
- The result will be a string following the JSON notation.

```
<script>
    varvalue = {
        name: "Logan",
        age: 21,
        location: "London"
    };
    varresult = JSON.stringify(value);
    document.write("value of result = "+ result + "<br>");
    document.write("type of result = "+ typeofresult);
</script>
```

Output

```
value of result = {"name":"Logan", "age":21, "location":"London"}
type of result = string
```

In JSON, date objects are not allowed. The JSON.stringify() function will convert any dates into strings.

```
const obj = {name: "John", today: new Date(), city : "New York"};
const myJSON = JSON.stringify(obj);
```

jQuery.parseJSON()

- JSON.parse() itself cannot execute functions or perform calculations. JSON objects can only hold simple data types and not executable code.
- This **parseJSON()** Method in jQuery takes a well-formed JSON string and returns the resulting JavaScript value.
  Var obj = jQuery.parseJSON('{ "name": "John" }');
  alert(obj.name);

JSON Comments

- JSON doesn't support comments! But as programmers, we are used to add comments wherever required.
- We can add a comment in JSON file included in the JSON object.

```
Var value = {
     _Comment1 : "This is student record",
     _Comment2 : "This is student record",

        name: "Logan",
        age: 21,
        location: "London"
    }
```

  We specify an underscore before comment to view it as a comment. But remember the comment is only a comment in the eyes of the developer but not the computer.
- We can even extract the comment's value from the JSON object.
  alert(value._comment);